定理証明的手法を用いた回路の自動合成 : 帰納方程式から回路への変換(アルゴリズムの数学的基礎理論とその応用)

AUTHOR(S):

原尾, 政輝; 岩沼, 宏治

CITATION:

原尾, 政輝 ...[et al]. 定理証明的手法を用いた回路の自動合成 : 帰納方程式から回路への変換(アルゴリズムの数学的基礎理論とその応用). 数理解析研究所講究録 1986, 591: 65-73

ISSUE DATE:

1986-05

URL:

http://hdl.handle.net/2433/99477

数理解析研究所講究録
591巻 1986年 65-73

定理証明的手法を用いた回路の自動合成
－帰納方程式から回路への変換－

原尾　政輝　　岩沼　宏治
Masateru HaRAO　　Kouji IWANUMA
（山形大学工学部　情報工学科）

あらまし　大規模・複雑化しているシステムの設計において，人手による方法は時間的制約や誤りの増加等の問題から不可能に近くなっている。そのため，計算機を用いた設計システム，ＣＡＤ，が研究・開発されて成果を上げている。特に，論理回路網のＣＡＤの分野ではシリコンコンパイラを最終目標として，出来るだけ高いレベルの記述からマスクパターンまでの，一貫した設計自動化が研究されている。

本論文では，高レベルの記述として帰納方程式を採用し，それから関数的記法に基づいた回路表現であるＦＮへの自動変換について考察する。先ず，回路網とＦＮとの関係について概観し，帰納方程式からＦＮへの変換規則の理論的性質について述べる。次いで，定理証明的手法による回路の自動合成手法について考察する。最後に，この手法に基づいた簡単な例を示す。

１．まえがき

回路，特にＶＬＳＩのように大規模な回路，の合成においては，設計者にとって分かり易くかつ記述し易い高レベルの言語で仕様記述を行い，その記述から詳細回路図レベルへ自動変換する設計自動化手法が重要な課題であり，シリコンコンパイラを始めいろいろ検討されている[7]。回路の自動設計では，素子の型に従って幾つかのレベルに分け，階層的設計を行う手法が有効である。本論文では，図１で示すようなレベルの階層を仮定する。これまでいろいろな自動合成システムの提案ないしは研究・開発が行われているが，その多くは機能レベルからの合成である。それはシステムレベルから回路への変換の過程には非決定的要素が多く，又作成された回路の完成度も人手によるものに比べてまだまだ問題があるからである。しかしJhonson[4]は，高レベル記述として帰納方程式を用い，その表現から回路への変換が有効な事を示している。本論文では，回路の表現としてデータフロー演算に基づく関数ネットワーク[2]（FN）を用い，帰納方程式からＦＮへの自動変換について考察する。　自動変換に関しては幾つかのの手法が開発されているが，上述の欠点を補った自動合成システムを構築するためには，変換過程で規則の適用や回路の簡単化などに関する熟練者の知識を内包し，推論や学習などの機能を実現出来るものが望ましい。定理明的手法は，このような知識工学的要求も満たしており，回路の自動合成にも有効な事が示されている[6,9]。このような考えに基づいたシステムの構成には，知識

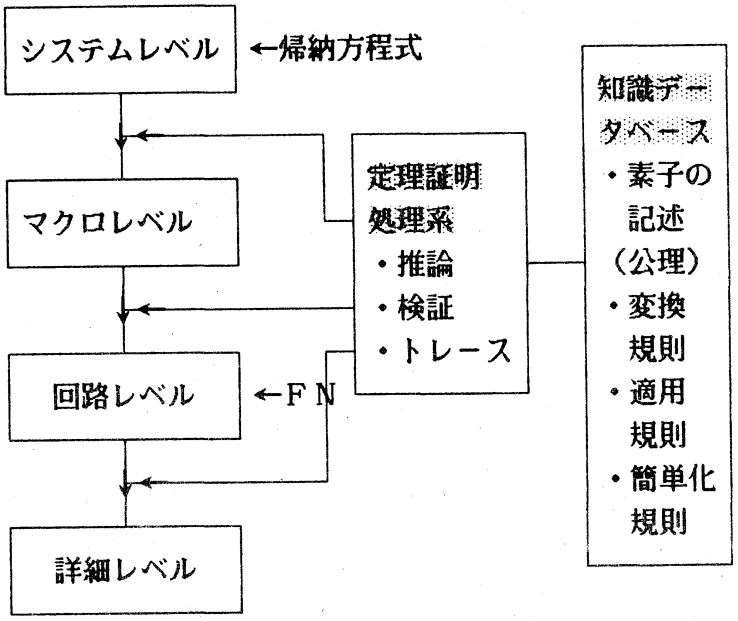


図１．知識工学的手法を用いた回路自動合成

データベースの作成に適し，推論や証明の機能を持った記述言語が必要である。本論文では，Prologが定理証明に基づいた処理系である事から，Prologを記述言語として用いる。本論文は，帰納方程式からFNへの知識工学的手法を用いた自動変換についての基本的性質を明らかにするのが目的である。第2章ではデータフロー演算に基づいた関数ネット(Functional Net)を定義し，その理論的性質について述べる。第3章では，回路の高位レベル記述である帰納方程式からＦＮへの変換規則について考察する。第4章では定理証明的手法に基づいた帰納方程式からＦＮへの自動変換の概要を示し，Prolog KABAを用いた実現例を示す。

２．回路とその表現

これまでにも種々のハードウェア記述言語が提案され実現されている。その大部分は，レジスタトランスファレベル以下のもので，シミュレーションを目的としたものである。本稿では，高レベルの記述として数学的表現である帰納方程式を採用し，それからマクロ素子レベル回路記述であるＦＮへの変換問題を考察する。ここでは細かいタイミングや信号の記述は行なわない。

２．１　関数ネット

回路の機能は，基本的にはデータフロー演算に基づいており，関数的表現と自然に対応する。ここで回路の素子は，関数演算子，述語演算子及びデータフロー制御のための数種の制御素子より成るとする。図２に，回路素子とその関数および Prolog の構文規則による関係型表現との対応を示している。

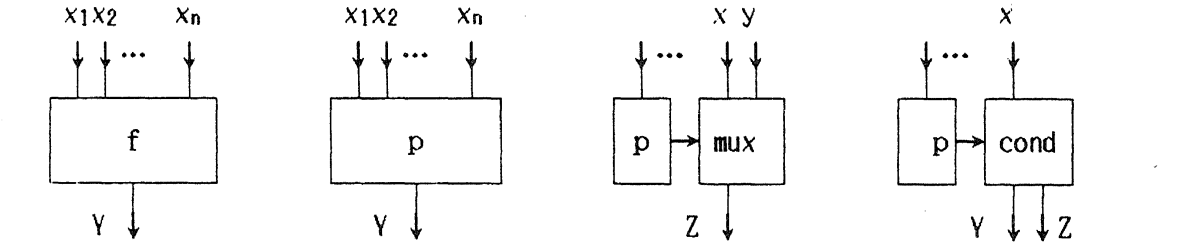


図2　関数ネットと回路演算子

２．２　合成・分解規則

一般の関数，述語演算子に対応する回路は，基本的な回路素子に原始帰納法で言う合成・代入の規則を用いて構成される。図３に基本的合成・分解規則を示す。

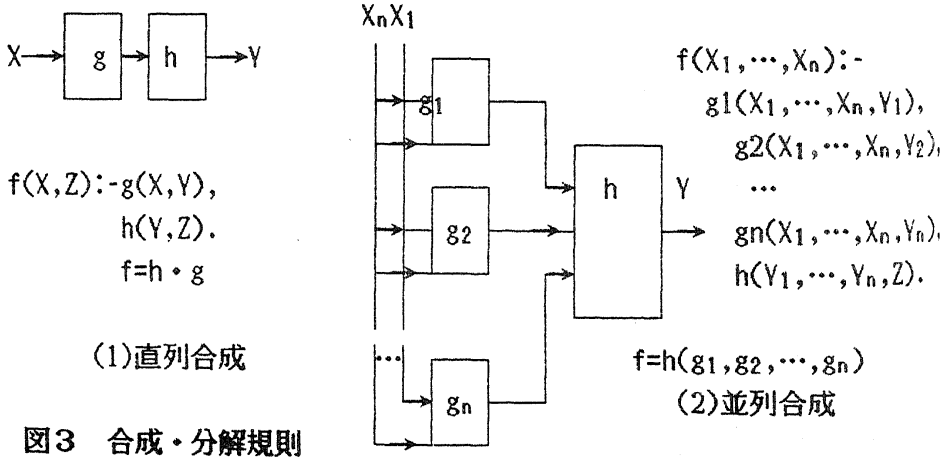


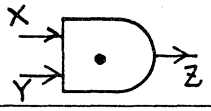
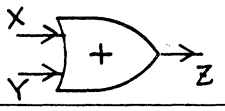
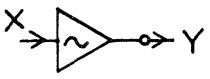
図３　合成・分解規則

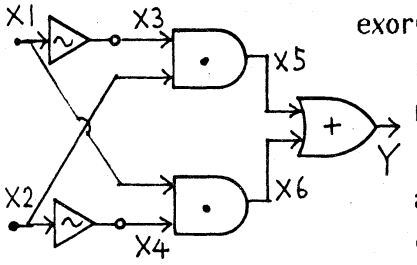
定義　回路の基本素子の集合をΩとする。Ωの要素を上述の規則に従って組合わせて構成される回路網を関数ネット(Functional Net)と呼ぶ.

2．3　ＦＮの意味記述

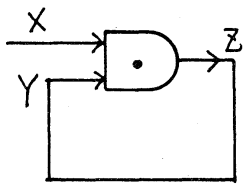
ＦＮの動作の検証・シミュレーションを行うためには，ＦＮの意味記述を明確にする必要がある。ＦＮが，データフロー制御に基づいて動作する回路網であると仮定すれば，記述には値，位置更には時間の情報も陽に表現出来る事が望ましい。検証手法の点からは，定理証明的手法が素直に導入出来るものが望ましい。以上の事より，記述は論理型言語を採用した。現在，空間・時間双方を記述出来る様相論理体系を定義し，論理型言語を拡張する事を計画している[3]。

<例１>　論理回路網の場合の基本素子集合Ωとその合成・分解規則の例を図３に示す。そこで，(a)は構文情報のみを(b)はより意味的情報を表現したものになっている。

素子			
(a)接続情報	and(X,Y,Z)	or(X,Y,Z)	not(X,Y)
(b)演算情報	and(1,1,1) and(X,0,0) and(0,Y,0)	or(0,0,0) or(X,1,1) or(1,Y,1)	not(0,1) not(1,0)




```
exor(X1,X2,Y):-
    not(X1,X3),
    not(X2,X4),
     and(X2,X3,X5),
    and(X1,X4,X6),
    or(X5,X6,Y).
```




```
func(X,Y,Z):-
 and(X,Y,Z),
(retract(value(Y,V));true),
 !,assert(value(Y,Z)).
```

図４　基本素子と回路記述

３．帰納方程式からＦＮへの変換

3．1　基本的性質

ＦＮをマクロ素子よりなる回路網とみなし，帰納方程式からＦＮへの変換アルゴリズムについて考える。回路網は，基本的には関数合成よりなるから反復構造を持つと言える。従って，この変換問題は帰納から反復への変換と捉らえることが出来る。理論的に良く知られているように末尾再帰のものは，反復表現出来るが，一般には帰納は反復には変換出来ない。　本稿では，帰納方程式を形により幾つかのクラスに分類し，その反復構造（ＦＮ）への変換問題を考察する。　特にスタックを素子として用いるならばすべての帰納方程式はＦＮへ変換出来る。

定数，項，変数，関数，述語，論理式…等の用語は通常の意味とする[5]。特に，個体変数をx,y,…，述語定数を p,q,…，関数定数をf,g,…，関数変数をF,G,…,等で表す。

定義　次の形で表される式を帰納方程式と呼ぶ：
F（x 1，…，x n）<=p 1→r 1，p 2→r 2，…，p n→r n．
特にp→r，¬p→qをp→r，qと表す。
<例2>Σ＝｛a，b｝とするときΣ*からΣ*への写像空間での帰納方程式の例を考える。
(1) last(x) <= eq?(tail(x),λ)→x, last(tail(x)).
(2) length(x) <= nil(x)→0, add$_1$(length (tail(x)).
(3) append(x,y) <= nil(y)→x,cat(x, tail(y)), head(y)).
(4) reverse(x) <= nil(x)→λ, cat(reverse(tail(x)),head(x)).
(5) mix(x) <= atom(x)→x, cat(mix(left(x)),mix(right(x))).
(6) ackermann(x,y) <= eq?(x,λ)→ay,
　　eq?(y,λ)→ackermann(tail(x),a),
　　ackermann(tail(x), ackermann(x,tail(y))).
ここで tail,nil,head,atom,left,right eq?などの関数は基本素子で実現されているとする[5]。

命題1　次の式は等価である。
(1) p(x) →r(x),s(x).
(2) mux(p(x),r(x),s(x)).
(3) for all x cond(p)(x) = [r,s](x). □

命題2　次の式は等価である。
(1) p→F(r_1,…,r_n),F(s_1,…,s_n).
(2) F([p→r_1,s_1],…,[p→r_n,s_n]). □

命題3　次の式は等価である。
(1) p→ F(r_1,…,r_n),F(s_1,…,s_n).
(2) F(mux(p,r_1,s_1),…,mux(p,r_n,s_n)). □

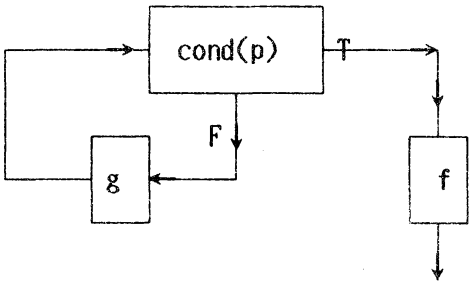


図5　基本帰納方程式とそのＦＮ

命題4　F,G に対して H が存在し，次の
関係が成り立つとする。
　　H(ω_1,r_1,…,r_n) = F(r_1,…,r_n),
　　H(ω_2,s_1,…,s_n) = G(s_1,…,s_n).
この時，次の式は等価である。
(1) p → F(r_1,…,r_n),G(s_1,…,s_n).
(2) H(mux(p,ω_1,ω_2),mux(p,r_1,r_2),…,mux(p,r_n,s_n)). □

定義　F(x) <= p(x) →f(x),F(g(x))を基本帰納方程式と呼ぶ。

命題5　基本帰納方程式は，図5のＦＮで実現される。

この回路を基本回路と呼ぶ。以下全ての帰納方程式が基本回路へ変換出来る事を示す。

3．2　基本帰納方程式への帰着可能性

関数変数が１個だけ出てくる帰納方程式を線形，そうでないものを非線形と呼ぶ。線形な方程式はスタックなしで基本帰納方程式へ変換出来る。非線形な方程式はスタックを用いることによって基本帰納方程式へ変換出来る。帰納方程式の形により４つのクラスを定義する。それぞれの代表的なものを表１に示している。　　表１

	クラス１	クラス２	クラス３	クラス４
方程式	G(x)<=p(x)→L(x),G(g(x)). L(x)<=q(x)→H(x),L(I(x)). … H(x)<=r(x)→i(x),H(h(x)).	F(x)<=p(x)→f(x), h(F(g_1(x),g_2(x)).	F(x)<=p(x)→f(x), h(Fg_1(x),Fg_2(x)).	それ以外の関数

例２において，(1)はクラス１，(2),(3),(4)はクラス２，(5)はクラス３，(6)はクラス４である。

命題６　クラス１の帰納方程式は，基本帰納方程式へ変換することが出来る。

（略証）コントロール・トークンωを導入して，次の形へまず変形する。

F(ω,x)<=eq?(ω,I)→i(x),
　　eq?(ω,G)→[p(x)→F(L,x),F(G,g(x))],
　　eq?(ω,L)→[q(x)→F(H,x),F(L,I(x))],
　　…
　　eq?(ω,H)→[r(x)→F(I,x),F(H,h(x))].

命題３，命題４を適用して最終的には次の形になる。

F(ω,x)<=eq?(ω,I)→i(x),F(f(ω,x))
f(ω,x)=(mux(eq?(ω,G),mux(p(x),L,G),mux(mux(eq?(ω,L),mux(q(x),H,L),mux(r(x),I,H))),
　mux(eq?(ω,G),mux(p(x),x,q(x)),mux(mux(eq?(ω,L),mux(q(x),x,I(x)),mux(r(x),x,h(x))))

ここで，F(G,x) = G(x).□

命題７(4)次のクラス２の帰納方程式は，基本帰納方程式へ変換することが出来る。

（１）　F(x) <= p(x)→f(x), h(F(g(x)))
（２）　F(x) <= p(x)→f(x), h(x, F(g(x))).

（略証）(1)まず，この式を次のような２変数関数に変形する。

G(x,y) <= p(x)→ H(y,f(x)), G(g(x),y),
H(x,y) <= p(x)→y, H(g(x),h(y)).

ここで　F(x) = G(x,x)に注意する。

(2)制御変数を導入して次の方程式に変換する事が出来る。

G(u,v,x,y,z) <= p(x)→ L(u, ,u, ,f(x)), G(u, ,g(x), ,).
L(u,v,x,y,z) <= p(x)→ M(u, g(x),g(x),u ,z).
M(u,v,x,y,z) <= p(x)→ L(u, ,v, ,h(y,z)),M(u,v,g(x),g(y),z).

このようにして得られた方程式はそれぞれクラス１の方程式の形であるから，命題６より変換可能で、
□

性質 クラス２の方程式には 次の変換規則が成り立つ。

(1)F(x) <= p(x)→f(x), h(F(g(x)))において，$fg^n(x)$がｎに依存しない値をとるとすると，この方程式は次のように変換出来る。

H(x,y) <= p(x)→y, H(g(x), h(y))).

(2)F(x) <= p(x)→f(x), h(F($g_1(x)$), $g_2(x)$) は，hが結合則を満たせば次のように変換出来る。

F(x) <= p(x)→ h(f(x), y), G($g_1(x)$, h($g_2(x)$,y) □

例３ 例２の方程式はそれぞれ次のように変換される。

(2)LEN(x) <= nil(x) →y, LEN(tail(x), $add_1(y)$))

(4)REV(x,y) <= nil(x)→cat(λ,y), REV(tail(x), cat(head(x), y)).

一般にクラス３およびクラス４の帰納方程式をＦＮに変換するには，基本素子としてスタックを新たに用いる。スタックは pop,push,top,empty? 等の関数・述語を用いて次のように定義される；

empty?(λ)=true.　empty?(push(u,v))=false

top(λ)=φ　　top(push(u,v))=u

pop(λ)=φ　　pop(push(u,v))=v

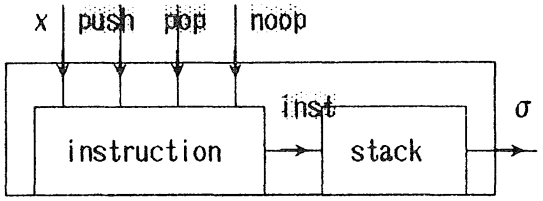


図６スタック素子

スタックを基本素子として用いるならば，次の性質が成り立つことは良く知られている。

命題９ クラス３の帰納方程式は，スタック１個を用いてＦＮに変換することが出来る。また，２個以上のスタックを用いれば全てのクラスの帰納方程式はＦＮに変換出来る。□

帰納方程式からスタックを含む反復構造への変換アルゴリズムについても幾つか提案されているが，一般的な有効なものは与えられていない。

＜例４＞ クラス３の帰納方程式(5)およびクラス４のAckermann関数(6)について，そのＦＮへの変換例を次に示す。

(5)mix(x,σ) <= atom?(x)→R(x,σ),mix(left(x),[a right(x)!σ]),

R(v,[ωz!σ']) <=empty?(σ)→v,

eq?(ω,0)→mix(z,[1,v!σ']),

eq?(ω,1)→R(cat(v,z),σ').

(2)ACKER(x,y,σ)<=eq?(σ,ε)→H(mux(eq?(x,λ),$ω_1$,$ω_2$),mux(eq?(x,λ),λ,x),

mux(eq?(x,λ),y,y),mux(eq?(x,λ),σ,σ)),

ACKER(mux(eq?(x,λ),top(σ),f_1),mux(eq?(x,λ),ay,f_2),mux(eq?(x,λ),pop(σ),f_3)).

但し，H($ω_1$,λ,y,ε) = ay,　H($ω_2$,x,y,ε) = ACKER(f_1,f_2,f_3)

f_1 = mux(eq?(y,λ),$tail_1$(x),$tail_1$(x)),

f_2 = mux(eq?(y,λ),1,$tail_1$(y)),

f_3 = mux(eq?(y,λ),σ,push($tail_1$(x),σ)).

4. 定理証明による自動合成

4.1 基本概念

論理の枠組みで用いられる定理証明的手法が，回路の自動合成にどのように応用出来るかについての基本的考え方を述べる。　ＦＮでは，cond とか muxのように関係として表わしたほうが容易かつ分かり易い素子を含んでる。また，述語表現は関係の記述にうまく適合する。このため本稿では，Prolog の構文に従った関係表現を用いる。回路の表現には，使用する素子と分解・合成に関する規則および規則の適用に関する規則等がある。以下それらの性質について概観する。

1)基本素子の表現：宣言的知識

構成要素となる素子を述語として表わす。これは，述語論理での公理に相当するもので，知識工学で言う宣言的知識に対応する。

2)合成・分解規則：手続き的知識

回路の変換に関する規則で，述語論理の推論に相当する。三段論法に従って，新しい基本素子を用いて実現可能な回路記述を作り出して行く。図７に大まかな考え方を示す。

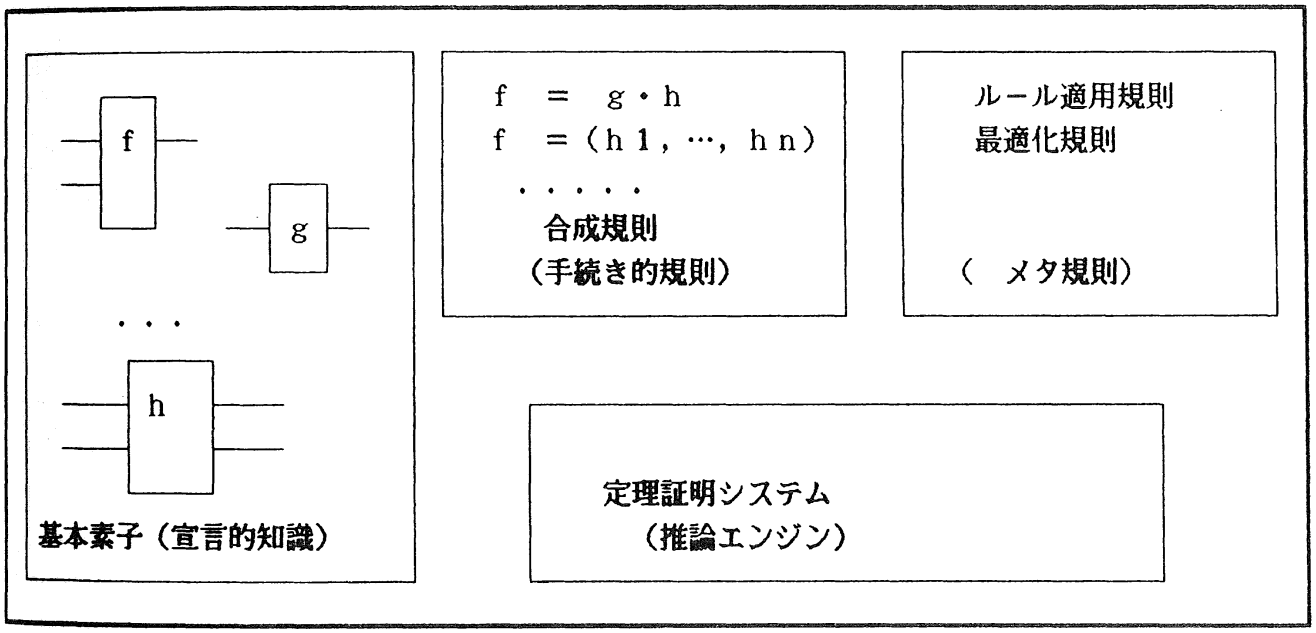


図７　定理証明による回路の自動合成

3)背理法による証明

合成したい回路の述語表現（仕様記述）をＱとする。また，使用可能な基本素子と合成・分解規則の述語表現を $\{p_i : i=1,\cdots,n\}$ で表わす。この時，基本素子の組合わせで回路Ｑが実現出来ることは，述語では次のように表わされる：

$$\left(\bigcap_{i=1}^{n} P_i\right) \rightarrow Q \qquad \cdots \quad (1)$$

(1)式の否定をとると，

$$\sim((\cap P_i) \rightarrow Q) = \sim(\sim(\cap P_i) \cup Q)$$

$$= (\cap P_i \cap Q) \qquad \cdots \quad (2)$$

(1)式が恒真式である事と(2)式が充足不能であることとは等価である。即ち，条件　～Qを仮定した時矛盾□が導かれたとき証明が完了する。この背理法による証明過程は，結論を出すに至るまでの可能な公理や規則の組合わせを機械的にさがしながら進む。このことは，回路Qを矛盾なく構成するために必要な素子とその接続情報を自動的に探し出している事に相当する。　例えば，結論Qを導く過程が図7のようであったとすれば，Qに至る履歴 p1,p2,…,Q が回路の接続情報を表わしている。

よく知られているように，Prolog　の処理系は，この背理法による証明に基づいて 作成されており，回路の仕様記述言語としてそのまま用いる。事が出来る。Prologを用いる場合には，(1)式に対応する証明すべき（合成すべき）回路記述を問の形　　?－　Q で与える。証明過程は，現在のところデバッガー，トレーサーを用いてチェックしている。それが回路合成の接続乗法を与える。又，公理（基本素子の述語表現）や問い合せ方法に工夫をすれば，回路動作の検証等にも拡張応用出来る。次に，その簡単な例を示す。

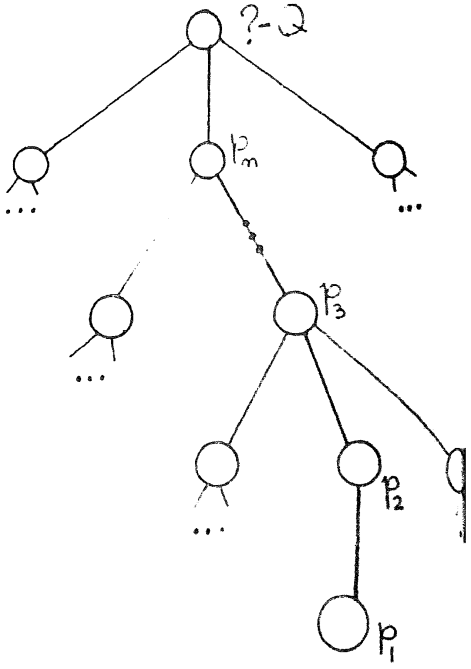
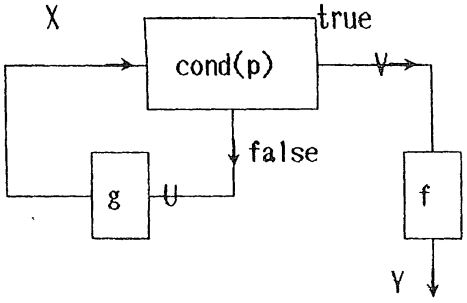


図8証明木

＜例4＞　基本帰納方程式の合成例。




```
f(X,Y).
g(X,Y).
cond(P,X,Y,Z).
func(P,f,g,X,Y):-cond(P,X,U,V),
                 g(U,X),f(V,Y).
```

```
!!?-trace.
yes
!!?-func(p,f,g,X,Y).
  [1] 0 Try   : func(p,f,g,X,Y) ?
Match : func(p,f,g,X,Y) :-
                 cond(p,X,U_1,V_1),
                 g(U_1,X),
                 f(V_1,Y).
  [2] 1 Try   : cond(p,X,U_1,V_1) ?
Match : cond(p,X,U_1,V_1).
  [2] 1 Succ  : cond(p,X,U_1,V_1)
  [3] 1 Try   : g(U_1,X) ?
Match : g(U_1,X).
  [3] 1 Succ  : g(U_1,X)
  [4] 1 Try   : f(V_1,Y) ?
Match : f(V_1,Y).
  [4] 1 Succ  : f(V_1,Y)
  [1] 0 Succ  : func(p,f,g,X,Y)
X       = X,
Y       = Y
yes
```

6. 結論

現在，2章，3章で述べた規則を宣言的知識および手続き的知識として組み込んだ簡単なシステムをProlog KABAを用いて作成している。その結果，Debugger やTracerを改良し接続情報をうまく取り出せるように工夫すれば，この手法である程度効果的に回路の自動合成が可能である。ただ，回路の動作や意味記述を行い回路の検証を行う場合には，ループ情報の取り扱いが本質的である。そのためには，記述言語の意味記述に対する記述能力を高める必要がある。筆者等は，時間や空間の記述が可能な様相論理に基づいたシステム記述言語を設計し，それを用いた回路自動合成システムの構成を最終目標として研究を進めている。また，このシステムを実際的なものにするには，推論規則の効果的適用法を工夫する必要がある。これには，メタ述語，制御規則の導入等が一つの有効な方法と思われる。

理論的には，ＦＮに変換可能な帰納方程式のクラスの特性化や変換アルゴリズムの設計等が上げられる。

謝辞：この研究の一部は，科学研究費補助金（一般研究（Ｃ）課題番号60580016）の援助の下に行われたものである。

文献

[1]青野，バイサン，原尾，野口：ハードウェアアルゴリズムの自動合成，東北大学談話回会記録、53巻第1号，1984.

[2]T.Ito:Regular Tree Expressions and Behaviors of Functional Nets,private communication.

[3]岩沼，原尾，野口：時空間様相論理の完全無矛盾な公理系，信学技報，オートマトンと言語，AL 84-52,1985.

[4]S.D.Jhonson:Synthesis of Digital Designs from Recursion Equations,ACM distinguished Dissertations, MIT press,1983.

[5]Z.Manna:Mathematical Theory of Computation,MacGraw-Hill Inc. 1974

[6]N.J.Nilsson:Principles of Artificial Intelligence,Tioga Publishing Co.1980

[7]J.D.Ullmann:Computational Aspects of VLSI,Computer Press,1984.

[8]上原：ＣＡＤにおけるProlog,情報処理，Vol.25 No.12,1984.

[9]W.S.Wojciechowski,A.S.Wojcik:Automated Design of Multiple-valued Logic Circuits by Automatic Theorem Proving Techniques,IEEE Trans.on Computers,Vol.C-32,No.9,1983.